Efficiëntie van raymarching in renderen

Taeke Roukema

Oktober 2022

Samenvatting

Renderen met raymarching is fucking lijp brooo. Iedereen zou het moeten doen eigenlijk.

Inhoudsopgave

1 Voorwoord

2 Inleiding

2.1 Introductie onderwerp

Renderen is overal. Als je je telefoon opent zie je allerlei gerenderde vormen. Bij het ontbijt zijn verpakkingen volgeprint met teksten die met de computer getekend zijn. Als je langs een bouwterrein loopt zie je hyperrealistische visualisaties van de architectuur. Moderne blockbuster-films zitten tegenwoordig bomvol CGI[1]. En er zijn al tientallen jaren films te zien die helemaal door de computer gemaakt zijn.

Voor Toy Story 3 (Figuur 2.1) werd er gemiddeld zeven uur over gedaan om een frame te renderen [Lehrer, 2010]. En dat terwijl er gebruik werd gemaakt van twee gigantische render farms[2]. Het renderen van films kost niet alleen enorm veel tijd, maar ook veel energie. Het is dus belangrijk dat het zo efficiënt mogelijk gebeurd. Er wordt over de hele wereld voortdurend onderzoek gedaan naar manieren om dit proces efficiënter te maken en te verbeteren. De opkomst van kunstmatige intelligentie begint al bewegingen te maken in de wereld van CGI [Anderson, 2021]. Maar er wordt ook voortdurend voortuitgang gemaakt op fundamentelere manieren. Zo zijn er de afgelopen vijf jaar GPU's[3] van Nvidia op de markt gekomen met ingebouwde support voor realtime raytracing[Alwani, 2018]. Door op het hardware niveau de chips zo te ontwerpen dat ze heel goed zijn in bepaalde berekeningen die gebruikt worden voor het simuleren van licht kunnen GPU's gebruikt worden om voormalig minutendurende processen meer dan zestig keer per seconde uit te voeren.

Figuur 2.1: Een frame uit Toy Story 3, aan de linkerkant worden geen lichtberekeningen gedaan, en aan de rechterkant wel.

[1] **C**omputer **G**enerated **I**magery

[2] Een computercluster speciaal gemaakt voor het renderen van CGI, de term was geïntroduceerd in de productie voor Bored Room[Clay, 1990]

[3] **G**raphical **P**rogramming **U**nit

Figuur 2.2: De videogame Minecraft kan gebruik maken van Nvidia GPU's om realtime lichtsimulaties te berekenen.

2.2 Relevantie

2.3 Onderzoeksvraag/deelvragen

3 Theorie

3.1 Wat is renderen?

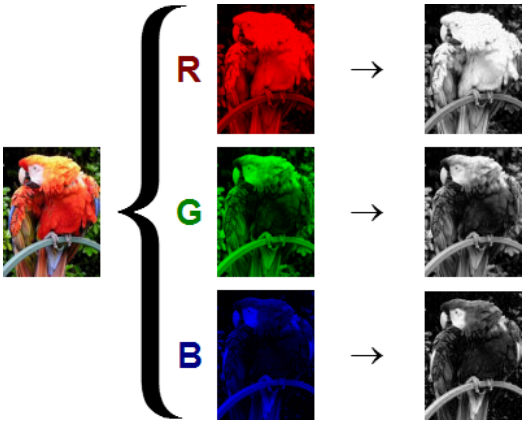
Renderen is, in feite, het weergeven van een representatie van een concept op een beeldscherm. Wij zijn voortdurend bezig met het interacteren met computers, en die interactie verloopt via het beeldscherm. Maar de computer kan uit zichzelf niet zomaar alles tekenen. Daar worden allemaal algoritmes voor geschreven. Een voorbeeld van zo'n algoritme is het tekenen van een rechthoek. In pseudocode zou je dat als volgt voor kunnen stellen:

```
function drawRectangle(x1, x2, y1, y2) {
        for (x = x1; x < x2; x++) {
                for (y = y1; y < y2; y++) {
                        drawPixel(x, y);
                }
        }
}
```

Het algoritme beschouwt elke pixel die binnen de rechthoek valt en kleurt die pixel. In dit geval wordt dat gedaan door twee loops, die samen alle mogelijke combinaties van x- en y-coördinaten doorlopen.

Renderen omvat, in principe, niks anders dan het aansturen van individuele pixels. Zo'n pixel heeft op de meeste moderne beeldschermen drie waarden die de kleur aansturen: R, G en B, die respectievelijk staan voor rood, groen en blauw. Ze kunnen een geheel getal tussen de 0 en 255 aannemen wat resulteert in 224 mogelijke kleuren.

Renderen doen we op een twee-dimensionaal beeldscherm. Dat betekent dat de positie van elke pixel te beschrijven is met twee waarden. Maar de wereld om ons heen kent niet twee, maar drie ruimtelijke dimensies. Door licht dat op ons netvlies valt na weerkaatst te zijn door verschillende objecten kunnen wij die wereld representeren in onze hersenen op een tweedimensionale manier. Camera's gebruiken een gelijksoortige techniek, de lens neemt het licht en projecteert het op een sensor die de intensiteit en de kleur waarneemt. Met het renderen van driedimensionale objecten proberen we deze processen na te bootsen.



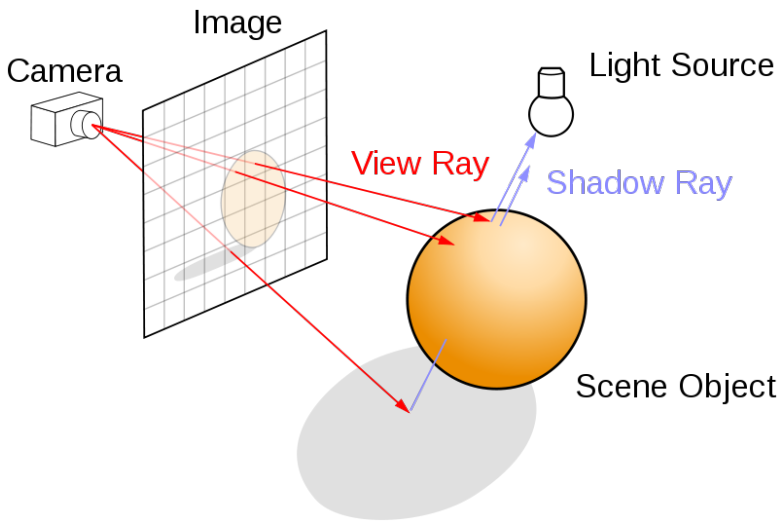
Figuur 3.1: De kleuren in een foto kunnen opgesplitst worden in rode, groene en blauwe kanalen.

3.2 Wat is raytracing?

Raytracing zou gezien kunnen worden als de meest voor de hand liggende rendermethode. Het ligt het dichtst in de buurt van het simuleren van echt licht. Het belangrijkste verschil tussen raytracen en licht in onze fysieke wereld is dat we met raytracen alleen het licht beschouwen wat zichtbaar is voor ons perspectief. Om dit te bereiken voeren we de lichtstralen niet af vanuit de lichtbron, maar vanuit de camera, vervolgens kaatsen we de straal af naar de lichtbron om te kijken hoe vel die plek zou zijn, en of er een ander object in de weg zit die een schaduw zou kunnen werpen. Op Figuur 3.2 is zichtbaar hoe dat raytracen in werking gaat.

3.3 Wat is rasterization?

3.4 Wat is raymarching?



Figuur 3.2: Een diagram die laat zien hoe raytracen werkt

4 Hypothese

5 Ontwikkeling

5.1 Hardware

5.2 Software

5.3 Programmeren

6 Methode

6.1 Variabelen

6.2 Meetmethoden

7 Resultaten

7.1 Snelheid

7.2 Geheugenbezetting

7.3 Renders

8 Nauwkeurigheidsanalyse

9 Conclusie

10 Discussie

11 Nawoord

Bedankt aan mijn moeder Arria Gosman.

12 Literatuurlijst

Referenties

[Alwani, 2018] Alwani, R. (2018). Microsoft and nvidia tech to bring photo-realistic games with ray tracing.

[Anderson, 2021] Anderson, M. (2021). Nerf moves another step closer to replacing cgi.

[Clay, 1990] Clay, J. (1990). Making of bored room (production).

[J.H. Reif and Yoshida, 1991] J.H. Reif, J. T. and Yoshida, A. (1991). Computability and complexity of ray tracing.

[Lehrer, 2010] Lehrer, J. (2010). How toy story 3 was made.

13 Logboek

Activiteit	Datum	Tijd (m)	Totale tijd (h)	% Voltooid
Programmeren	20220906	45	0.8	0.94%
Programmeren	20220908	30	1.3	1.56%
Gesprek met begeleider	20220909	20	1.6	1.98%
Programmeren	20220921	90	3.1	3.85%
Inhoudsopgave Opzet	20220928	35	3.7	4.58%
Schrijven Theorie/Achtergrond Renderen	20220930	45	4.4	5.52%
Opzetten LaTeX Document	20221002	120	6.4	8.02%